

Carnegie Mellon
Software Engineering Institute

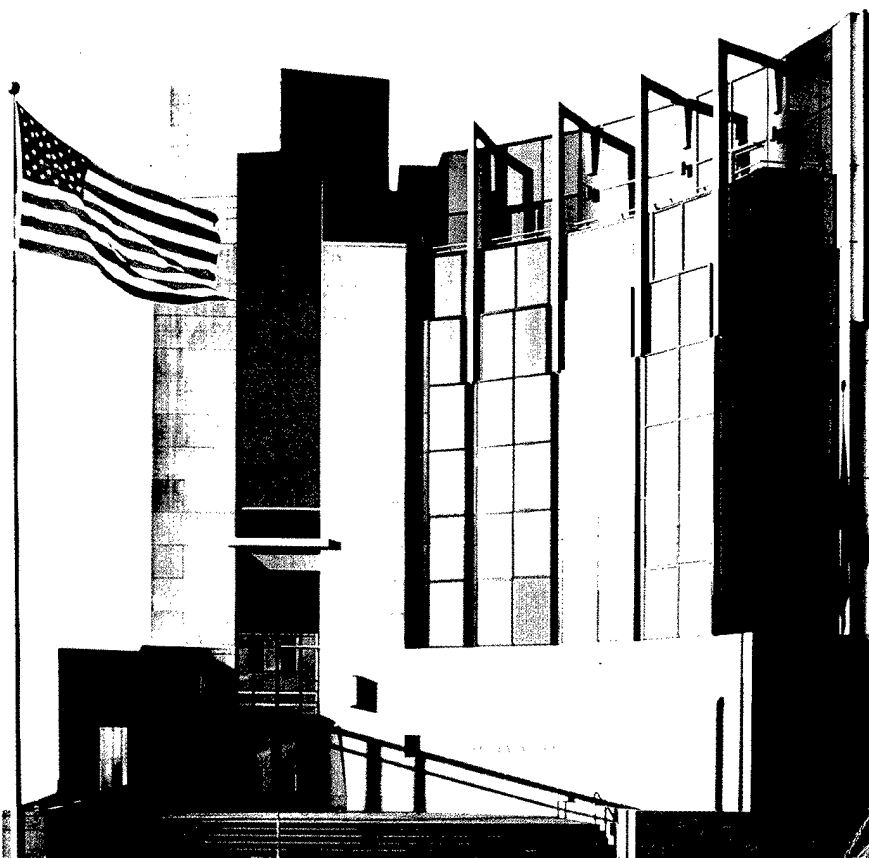
The Team Software ProcessSM (TSPSM)

Watts S. Humphrey

November 2000

TECHNICAL REPORT
CMU/SEI-2000-TR-023
ESC-TR-2000-023

20010312 119



Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of "Don't ask, don't tell, don't pursue" excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.



Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

The Team Software ProcessSM (TSPSM)

CMU/SEI-2000-TR-023
ESC-TR-2000-023

Watts S. Humphrey

November 2000

Team Software Process Initiative

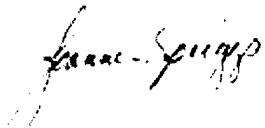
Unlimited distribution subject to the copyright.

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Joanne E. Spriggs
Contracting Office Representative

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 2000 by Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. Requests for permission to reproduce this document or prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (<http://www.sei.cmu.edu/publications/pubweb.html>).

Table of Contents

Acknowledgements	vii
Abstract	ix
1 Software Quality	1
2 How the TSP Was Developed	3
2.1 Engineering Teamwork	3
2.2 The Conditions for Teamwork	4
2.3 Effective Teams	5
2.4 Building Effective Teams	6
3 An Operational Team Process	7
4 The Structure of the TSP	9
5 Launching a TSP Team	11
6 The TSP Teamworking Process	15
6.1 Leading the Team	15
6.2 Communication	16
6.3 Maintaining the Plan	16
6.4 Rebalancing Team Workload	18
7 TSP Quality Management	19
7.1 The Quality Plan	19
7.2 Identifying Quality Problems	23
7.3 Finding and Preventing Quality Problems	26
8 TSP Introduction	29
9 TSP Experience	31
10 Status and Future Trends	33
10.1 Training and Support	33
10.2 Future Trends	33

List of Figures

Figure 1: Process Improvement Methods	8
Figure 2: TSP Team-Building	9
Figure 3: The TSP Process Flow	10
Figure 4: The TSP Launch Process	11
Figure 5: Percent Defect Free (PDF)	24
Figure 6: Defect Removal Profile	25
Figure 7: The Quality Profile	26
Figure 8: TSP Test Defects	32
Figure 9: TSP Test Time	32

List of Tables

Table 1:	The TSP Team Launch—Script LAU	12
Table 2:	Weekly Team Data	17
Table 3:	TSP Quality Guidelines	20
Table 4:	TSP Quality Plan—Form SUMQ	21

Acknowledgements

Contributing reviewers for this article were Eileen Forrester, Marsha Pomeroy Huff, Alan Koch, Don McAndrews, Jim McHale, Julia Mullaney, Mark Paulk, Bill Peterson, and Janice Marchok Ryan.

Abstract

The Team Software ProcessSM (TSP) guides engineering teams in developing software-intensive products. Early experience with the TSP shows that its use improves the quality and productivity of engineering teams while helping them to more precisely meet cost and schedule commitments. The TSP is designed for use with teams of 2 to 20 members, and the larger multi-team TSP process is designed for teams of up to about 150 members. While TSP versions are planned for larger projects, they are not available at the time of this writing.

This report describes the TSP and how it was developed. Starting with a brief background discussion of software quality, the report provides an overview of the basic elements of teamwork. It then describes the relationships among the TSP, Personal Software ProcessSM (PSP), and Capability Maturity Model[®] (CMM) process improvement initiatives. The report also describes the TSP process structure, launching a TSP team, the TSP teamworking process, and the issues and methods for introducing the TSP. The report concludes with a review of TSP experience, current status, and trends.

SM Team Software Process, TSP, Personal Software Process, and PSP are service marks of Carnegie Mellon University.

[®] Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.

1 Software Quality

Team Software Process (TSP) development follows the quality strategy that was originated by W. Edwards Deming and J.M. Juran [Deming 82, Juran 88]. This strategy was extended to the software process by Michael Fagan in 1976 [Fagan 76, Fagan 86]. It was further extended with the introduction of the Capability Maturity Model (CMM) in 1987 and the Personal Software Process (PSP) in 1995 [Humphrey 89, Humphrey 95, Paulk 95].

Following the PSP, a further important step in software process improvement was the introduction of the Team Software Process (TSP). The TSP provides a disciplined context for engineering work. The principal motivator for the development of the TSP was the conviction that engineering teams can do extraordinary work, but only if they are properly formed, suitably trained, staffed with skilled members, and effectively led. The objective of the TSP is to build and guide such teams.

2 How the TSP Was Developed

In 1996, Watts Humphrey developed the initial version of the TSP process. His objective was to provide an operational process to help engineers consistently do quality work. He designed the initial TSP0 process to be as simple as possible, tried it with two teams, and then reviewed the results to see how it worked. He then identified where the teams needed further guidance and enhanced the process to provide that guidance. The first TSP0 process was designed for PSP-trained teams who received no training or guidance other than that provided by the TSP process and the team's immediate management.

Based on the results from the two initial TSP teams, it was clear that the TSP helped engineers to do disciplined work but that more guidance and support was needed. It was also obvious that management must broadly support the TSP process. An enhanced TSP0.1 process was then used by additional teams, providing more information on needed process refinements.

Over the next three years, Humphrey developed nine more TSP versions. At the beginning, his objective was to see if a general-purpose team process could help engineering teams to do their work. Once it was clear that the TSP met this basic objective, his TSP process development efforts were directed toward simplifying the process, reducing its size, and providing the support and guidance needed to make the process most efficient and useful. As a result, the most recent TSP versions are substantially smaller than the TSP0.1 and TSP0.2 versions developed in late 1996 and early 1997.

As more groups have used the TSP process, several introduction methods have been developed to assist engineers and managers in building teams, better following the process, and periodically reassessing and replanning projects. Various prototype support tools have also been developed to simplify the engineers' planning, data recording, data analysis, and project reporting activities.

2.1 Engineering Teamwork

Teams are required for most engineering projects. Although some small hardware or software products can be developed by individuals, the scale and complexity of modern systems is such, and the demand for short schedules so great, that it is no longer practical for one person to do most engineering jobs. Systems development is a team activity, and the effectiveness of the team largely determines the quality of the engineering.

There are different kinds of teams. In sports, for example, a basketball team's positions are dynamic while baseball team members have more static roles. However, in both cases the members must all work together cooperatively. Conversely, wrestling and track teams are composed of individual competitors who do not dynamically interact, although the members support each other socially and emotionally.

In engineering, development teams often behave much like baseball or basketball teams. Even though they may have multiple specialties, all the members work toward a single objective. However, on systems maintenance and enhancement teams, the engineers often work relatively independently, much like wrestling and track teams.

A team is more than just a group of people who happen to work together. Teamwork takes practice and it involves special skills. Teams require common processes; they need agreed-upon goals; and they need effective guidance and leadership. The methods for guiding and leading such teams are well known, but they are not obvious. The Software Engineering Institute (SEI) is supporting the TSP as a way to guide engineers and their managers in using effective teamwork methods.

2.2 The Conditions for Teamwork

A team is a group of people who share a common goal. They must all be committed to this goal and have a common working framework. The following definition for a team has been adapted from [Dyer 84]:

- A team consists of at least two people.
- The members are working toward a common goal.
- Each person has a specific assigned role.
- Completion of the mission requires some form of dependency among the group members.

The four parts of this definition of a team are all important. For example, it is obvious that a team must have more than one member, and the need for common goals is also generally accepted. However, it is not as obvious why team members must have roles. Roles provide a sense of ownership and belonging. They help to guide team members on how to do their jobs; they prevent conflicts, duplicate work, and wasted effort; and they provide the members with a degree of control over their working environment. Such a sense of control is a fundamental requirement for motivated and energetic team members.

Interdependence is also an important element of teamwork. It means that each team member depends to some degree on the performance of the other members. Interdependence improves individual performance because the members can help and support each other. For example, design teams generally produce better designs than any individual member could have produced alone. This is because the team members have a broader set of skills and experiences than any one of the members has alone. Team performance is further enhanced by the social

support of membership. Human beings are social animals and few people like to work entirely by themselves—at least not for very long. Because of the social context of teams, the members will generally make a special effort to meet their obligations to the rest of the team. Through mutual support and interdependence, teams become more than just the sum of their individual members.

2.3 Effective Teams

To be effective, teams must be properly skilled and be able to work as cohesive units. Effective teams have certain common characteristics:

- The members are skilled.
- The team's goal is important, defined, visible, and realistic.
- The team's resources are adequate for the job.
- The members are motivated and committed to meeting the team's goal.
- The members cooperate and support each other.
- The members are disciplined in their work.

Another characteristic of effective teams is their ability to innovate. Innovation is more than just thinking up bright ideas; it requires creativity and a lot of hard work. Just about every engineering task is part of an innovative endeavor. Innovative teams must have skilled and capable people who are highly motivated. They must be creative, flexible, and disciplined. They must strive to meet demanding schedules while adjusting to changing needs. They must also control costs and schedules while keeping management informed of their progress.

To be innovative and effective, engineering teams must work in a trusting and supportive environment [Shellenbarger 00]. Engineering teams are composed of extremely capable people who can quickly sense a lack of trust. When managers do not trust their teams to make aggressive schedules or to strive to meet these schedules, the engineers will know it. When engineers do not feel trusted and respected, they often feel antagonized and manipulated. These engineers no longer feel loyal to the organization and can easily lose their commitment to the team.

Since people generally work harder when they face an important and meaningful challenge, it is appropriate for management to challenge their teams with aggressive goals. But when the teams respond to the challenge with a plan, management must be willing to negotiate realistic commitments that the engineers believe they can meet. Few people will work diligently to meet a seemingly hopeless schedule. To perform effectively, teams must believe that their project is important and that the schedule is achievable.

2.4 Building Effective Teams

The TSP is designed to establish the conditions that characterize effective teams [Cummings 87, DeMarco 87, Dyer 84, Katzenbach 93, Mohrman 95, Shaw 81, Stevens 94]. The team-building principles used in the TSP to establish these conditions are as follows:

- The team members establish common goals and defined roles.
- The team develops an agreed-upon strategy.
- The team members define a common process for their work.
- All team members participate in producing the plan, and each member knows his or her personal role in that plan.
- The team negotiates the plan with management.
- Management reviews and accepts the negotiated plan.
- The team members do the job in the way that they have planned to do it.
- The team members communicate freely and often.
- The team forms a cohesive group: the members cooperate, and they are all committed to meeting the goal.
- The engineers know their status, get feedback on their work, and have leadership that sustains their motivation.

Effective team formation requires that the members truly understand what they are supposed to do, agree on how to do the job, and believe that their plan is achievable. These conditions can all be established by involving the engineers in producing their own plans. Then, assuming that these plans are competently made, teams can almost always sell their plans to management.

While all these conditions are necessary for effective teamwork, the specific ways for establishing these conditions are not obvious. The TSP provides the explicit guidance that organizations need to build effective engineering teams.

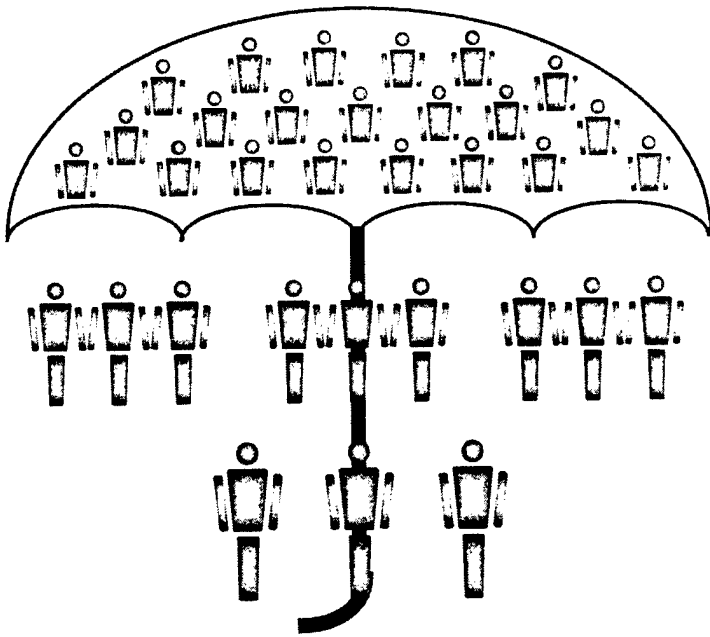
3 An Operational Team Process

To do disciplined work, engineers need what Deming calls “operational processes” [Deming 82]. These are processes that define precisely how the work is to be done. While most poorly defined software processes are large and comprehensive text descriptions that are filed in process definition books, an operational process is more like a script. It is designed to be used by the team members when they do the work.

The TSP provides a defined operational process to guide engineers and managers through the team-building steps. This process specifies the steps needed to establish an effective team-working environment. Without specific guidance, engineers must work out the details of team-building and teamworking for themselves. Since defining these details involves considerable skill and effort, and since few engineers have the experience or time to work out all of the necessary details, engineering teams generally follow ad-hoc team-building and team-work processes. This wastes time and it often produces poorly functioning teams.

With a defined process and a plan that follows that process, engineers can be highly efficient. If they don't have such a process, they must stop at each step to figure out what to do next and how to do it. Most engineering processes are quite complex and involve many steps. Without specific guidance, engineers are likely to skip steps, to do steps in an unproductive order, or to waste time figuring out what to do next. The TSP provides the operational processes needed to form engineering teams, to establish an effective team environment, and to guide teams in doing the work.

As shown in Figure 1, the TSP is one of a series of methods that can help engineering teams to more effectively develop and support software-intensive systems. The Capability Maturity Model (CMM) provides the overall improvement framework needed for effective engineering work [Paulk 95]. The Personal Software Process (PSP) provides the engineering disciplines that engineers need for consistently using a defined, planned, and measured process [Humphrey 95]. The TSP couples the principles of integrated product teams with the PSP and CMM methods to produce effective teams. In essence, the CMM and PSP provide the context and skills for effective engineering while the TSP guides engineers in actually doing the work. Thus, the TSP capitalizes on the preparation provided by the PSP and CMM, while also providing explicit guidance on how to do the work.



CMM - Improves organization's capability; management focus.

TSP - Improves team performance; team and product focus.

PSP - Improves individual skills and discipline; personal focus.

Figure 1: *Process Improvement Methods*

4 The Structure of the TSP

The principal elements of the TSP process are shown in Figure 2. Before the members can participate on a TSP team, they must know how to do disciplined work. As shown in this figure, training in the Personal Software Process (PSP) is required to provide engineers with the knowledge and skills to use the TSP. PSP training includes learning how to make detailed plans, gathering and using process data, developing earned value plans, using earned value to track a project, measuring and managing product quality, and defining and using operational processes. Engineers must be trained in these skills before they can participate in TSP team building or follow the defined TSP process.

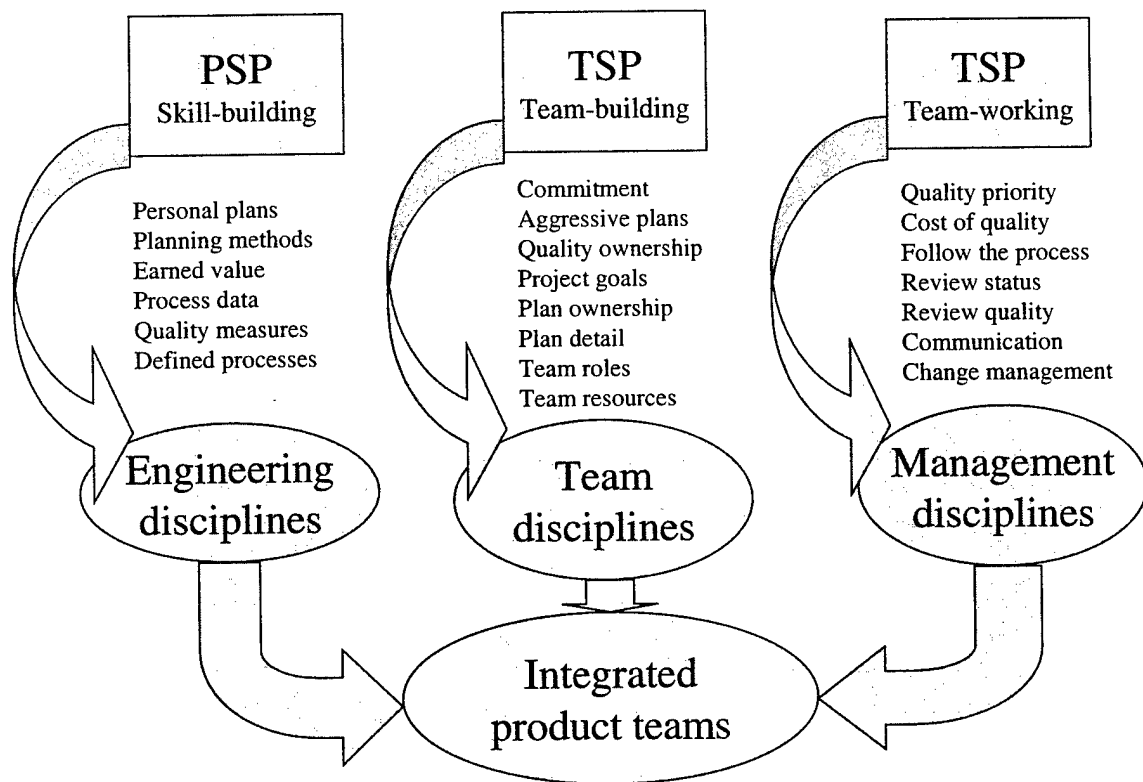


Figure 2: TSP Team-Building

While there are many ways to build teams, they all require that the individuals work together to accomplish some demanding task. In the TSP, this demanding team-building task is a four-

day planning process that is called the team launch. In a launch, all the team members develop the strategy, process, and plan for doing their project. After completing the launch, the team follows its own defined process to do the job.

As shown in Figure 3, TSP teams are relaunched periodically. Because the TSP process follows an iterative and evolving development strategy, periodic relaunches are necessary so that each phase or cycle can be planned based on the knowledge gained in the previous cycle. The relaunch is also required to update the engineers' detailed plans, which are usually accurate for only a few months. The reason for having a relaunch is that detailed plans can only be accurate for a few months. In the TSP launch, teams make an overall plan and a detailed plan for about the next three to four months. After the team members have completed all or most of the next project phase or cycle, they revise the overall plan if needed and make a new detailed plan to cover the next three to four months. They are guided in doing this by the TSP relaunch process.

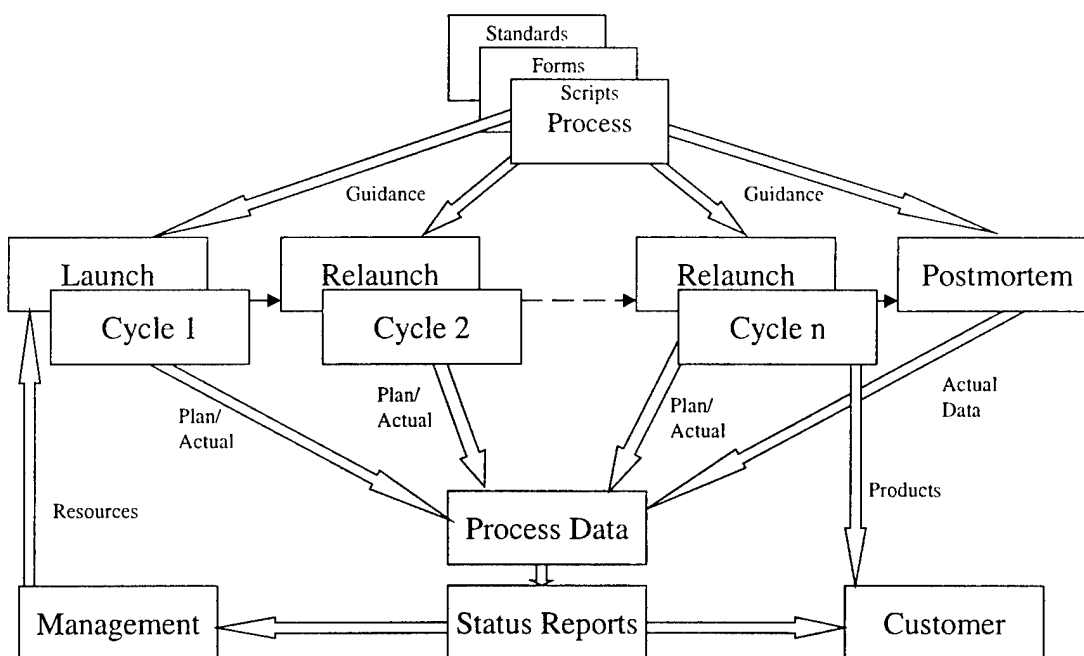


Figure 3: The TSP Process Flow

5 Launching a TSP Team

Once the team members have been properly trained and the team has been formed, the entire team participates in the TSP team launch. The launch process is shown in the launch script in Table 1 and in Figure 4. Each of the 9 launch meetings has a script that describes the activities in enough detail so that a trained launch coach can guide the team through the required steps. By following the launch process, teams produce a detailed plan. To become a cohesive and effective working unit, all the team members must be committed to the plan. To build this commitment, the TSP involves all the team members in producing that plan. Thus, by completing the TSP launch process, all the team members will have participated in producing the plan and they will all agree with and be committed to the plan that they produced.

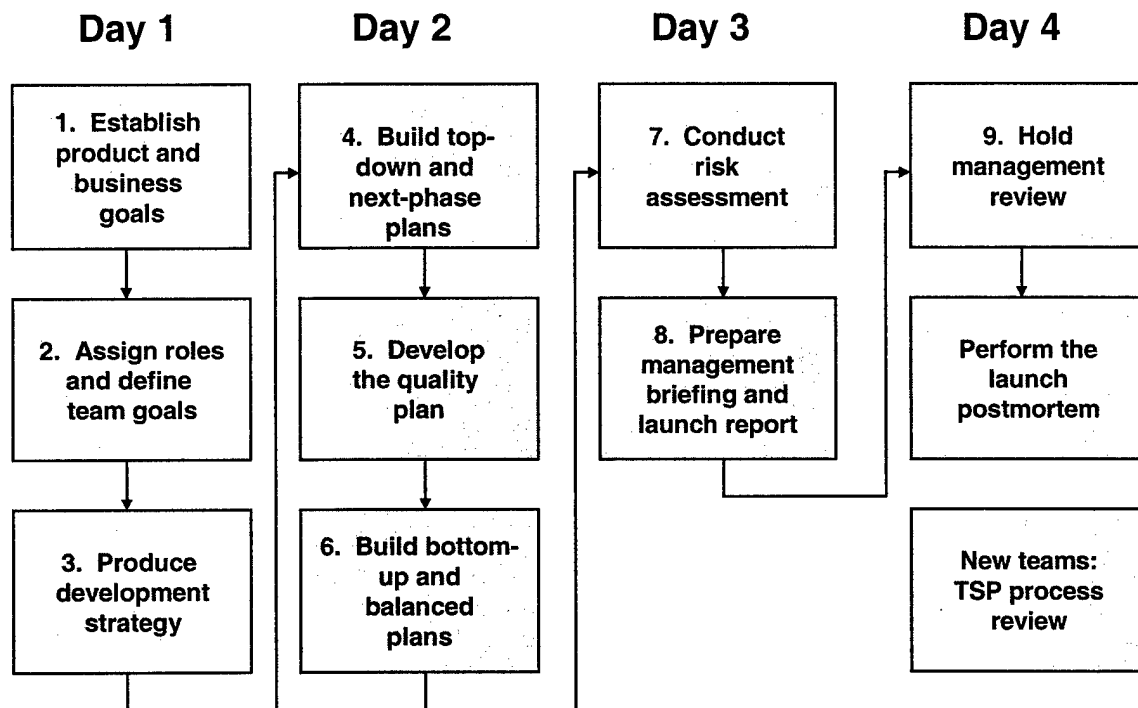


Figure 4: The TSP Launch Process

Table 1: The TSP Team Launch—Script LAU

Purpose	To guide integrated teams in launching a software-intensive project	
Entry Criteria	<ul style="list-style-type: none"> - The launch preparation work has been completed (PREPL, PREPT). - For the launch, the management and marketing representatives are prepared and available for meetings 1 and 10. - All team members and the team leader are committed to attend the launch meetings 1 through 9 and the postmortem. - An authorized launch coach is on hand to lead the launch process. 	
General	Timing <ul style="list-style-type: none"> - Meetings 1, 2, and 3 are held on launch day 1. - Meetings 4, 5, and 6 are held on day 2. - Meetings 7 and 8 are on day 3. - Meeting 9 and the launch postmortem are held at the close of day 3 or in the morning of day 4. 	
Step	Activities	Description
1	Project and Management Objectives	Hold team launch meeting 1 (use script LAU1). <ul style="list-style-type: none"> - Review the launch process and introduce team members. - Discuss the project goals with management and ask questions.
2	Team Goals and Roles	Hold team launch meeting 2 (use script LAU2). <ul style="list-style-type: none"> - Select team roles and backup roles. - Define and document the team's goals.
3	Project Strategy and Support	Hold team launch meeting 3 (use script LAU3). <ul style="list-style-type: none"> - Produce a system conceptual design and fix list (if needed). - Determine the development strategy and products to produce. - Define the development process to be used. - Produce the process and support plans.
4	Overall Plan	Hold team launch meeting 4 (use script LAU4). <ul style="list-style-type: none"> - Develop size estimates and the overall plan.
5	Quality Plan	Hold team launch meeting 5 (use script LAU5). <ul style="list-style-type: none"> - Develop the quality plan.
6	Balanced Plan	Hold team launch meeting 6 (use script LAU6) and produce <ul style="list-style-type: none"> - allocation of work to team members - bottom-up next-phase plans for each team member - a balanced next-phase plan for the team and each team member
7	Project Risk Analysis	Hold team launch meeting 7 (use script LAU7). <ul style="list-style-type: none"> - Identify and evaluate project risks. - Define risk assessment checkpoints and responsibilities. - Propose mitigation actions for immediate high-impact risks.
8	Launch Report Preparation	Hold team launch meeting 8 (use script LAU8). <ul style="list-style-type: none"> - Prepare a launch report to management.
9	Management Review	Hold team launch meeting 9 (use script LAU9). <ul style="list-style-type: none"> - Review launch activities and project plans with management. - Discuss project risks, responsibilities, and planned actions.
PM	Launch Post-mortem	Hold team launch meeting 9 (use script LAU9). <ul style="list-style-type: none"> - Walk through the weekly report preparation. - Gather launch data and produce a launch report. - Enter this report in the project notebook. - Assess the launch process and prepare PIPs.
Exit Criteria		<ul style="list-style-type: none"> - Launch completed with documented team and team member plans - Defined team roles, goals, processes, and responsibilities - Management agreement with the team plan or resolution actions identified and responsibilities assigned - Launch data filed in the project notebook (NOTEBOOK spec.)

Teams generally need professional guidance to properly complete the launch process. This guidance is provided by a trained launch coach who leads the team through the launch process. While the TSP scripts provide essential guidance, every team has unique problems and issues, so a simple process cannot possibly provide all the material needed to guide an inexperienced team through the launch process. Unless teams are very experienced and have a team leader who has completed several TSP projects, they generally need the support of a trained launch coach.

In launch meeting 1, the team, team leader, and launch coach meet with senior management and marketing representatives. Senior management tells the team about the project, why it is needed, the reasons for starting the project, and management's goals for the project. The marketing representative explains the marketing need for the product, any important competitive concerns, and any special customer considerations that the team needs to know. The objective of meeting 1 is to inform all the team members about the job, to describe management's goals for the team, and to convince the team members that management is relying on them to do this important project.

In launch meetings 2 through 8, the team, team leader, and coach meet with no observers or visitors. During these meetings, the team is led through a series of steps that are designed to produce the conditions for effective teamwork.

In launch meeting 2, the team documents its goals and selects the team member roles. The standard TSP team roles are team leader, customer interface manager, design manager, implementation manager, test manager, planning manager, process manager, quality manager, and support manager. Other possible roles can be assigned as needed. Examples would be safety manager, security manager, or performance manager. Every team member takes at least one team role. When there are more than eight team members, roles can be added or some engineers can serve as assistant role managers. The team leader generally does not take any other role.

In launch meetings 3 and 4, the team makes the overall project strategy and plan. The engineers produce a conceptual design, devise the development strategy, define the detailed process they will use, and determine the support tools and facilities they will need. They list the products to be produced, estimate the size of each product, and judge the time required for each process step. Once the tasks have been defined and estimated, the engineers estimate the hours that each team member will spend on the project each week. From the task estimates and weekly hours, the team generates the schedule.

Once they have an overall plan, the engineers produce the quality goals and plan in launch meeting 5. This plan both defines the quality actions the team plans to take, and it provides a measurable basis for tracking the quality of the work as it is done. In making the quality plan, the team members estimate the number of defects they will inject and remove in each phase, and how many defects will be left for system test, customer acceptance testing, and final

product delivery. Next, in meeting 6, the team members make detailed next-phase plans and then review the entire team workload to ensure that these plans evenly distribute the tasks among the members. The result is then what is called a balanced team plan. During meeting 7, the engineers identify the major project risks and rank them for likelihood and impact. The team also assigns a team member to track each risk and it prepares a mitigation plan for the most significant risks.

After the team has completed its plan, the members hold meeting 8 to prepare for the management review and then they conduct the review with management in meeting 9. During this meeting, the team explains the plan, describes how it was produced, and demonstrates that all the members agree with and are committed to the plan. If the team has not met management's objectives, it should generally prepare and present alternate plans that show what could be done with added resources or requirements changes. The principal reason for showing alternate plans is to provide management with options to consider in case the team's plan does not meet business needs. At the end of the TSP launch, the team and management should agree on how the team is to proceed with the project.

In the final postmortem step, the team reviews the launch process and submits process improvement proposals (PIPs) on suggested process improvements. The team also gathers and files the launch data and materials for later use.

6 The TSP Teamworking Process

Once the TSP team is launched, the principal need is to ensure that all team members follow the plan. This includes the following major topics:

- Leading the team
- Process discipline
- Issue tracking
- Communication
- Management reporting
- Maintaining the plan
- Estimating project completion
- Rebalancing team workload
- Relaunching the project
- TSP quality management

6.1 Leading the Team

The team leader is responsible for guiding and motivating the team members, handling customer issues, and dealing with management. This includes the day-to-day direction of the work, protecting team resources, resolving team issues, conducting team meetings, and reporting on the work. Overall, the team leader's principal responsibility is to maintain the team's motivation and energy and to ensure that it is fully effective in doing its work.

One key leadership responsibility is maintaining process discipline. Here, the team leader ensures that the engineers do the job the way they had planned to do it. During the launch, they defined the process for doing the job. While doing the job, the team leader monitors the work to ensure that everyone follows the process and plan that the team produced.

Almost every project faces heavy schedule and resource pressure, so there is always a temptation to cut corners. However, when teams stop following their defined processes, they have no way to tell what they are supposed to do or where they stand on the job. In monitoring process discipline, the team leader should check that every team member records his or her process data, reports on weekly status, and produces quality products.

Another important team leader responsibility is ensuring that all of the issues that the team members identify are managed and tracked. With the TSP, engineers generally discuss problems in the weekly team meeting. The team leader should first check that each issue is one that the team should handle and if it is, decide which team member should be responsible for managing and tracking it. Finally, the team tracks every issue with the issue tracking log (ITL) and reviews all the outstanding issues at each weekly meeting.

6.2 Communication

The team leader is responsible for maintaining open and effective team communication. When team members do not know the project's status, understand what their team mates are doing, or know what challenges lie ahead, it is hard for them to stay motivated. Communication is a key part of maintaining the team's energy and drive and facilitating communication is a key part of the team leader's responsibilities.

During the weekly meeting, the team leader first reviews project status and any management issues or concerns. The team members then each review their work for the previous week, the work they plan for the next week, their role-management activities, and the status of the risks they are tracking. They also bring up any issues or problems and describe any areas where they will need help or support during the next week.

Another critical team leader responsibility is keeping management informed about team status and progress. The TSP process calls for teams to make weekly reports that show where the team stands against the plan. The process also calls for frequent, factual, and complete customer status reports.

6.3 Maintaining the Plan

Once teams have completed the project launch and started on the job, the plan guides the work. It also provides a benchmark for measuring progress as well as means to identify problems that might threaten the project schedule. With sufficient warning, teams can often take timely action to prevent schedule slippage.

TSP teams track progress against the plan every week using a method called *earned value* [Humphrey 95]. With earned value, each task is assigned a value based on the percentage of the total project estimate that is required for that task. Thus, if a project was planned to take 1,000 task hours, a 32-hour task would have 3.2 planned value, or $100 \times 32 / 1000 = 3.2\%$. Then, when the team has completed that task, the engineers would have accumulated 3.2 earned value points, no matter how long the task actually took.

Engineering teams have many types of tasks and on any reasonably complex project, the tasks will often be completed in a different order than originally planned. Since some tasks will be completed early and others will be late, there is no simple way to tell whether the project is ahead of schedule or behind. The earned value method provides a value for every

task, and when that task is completed, the team earns that value. Thus, with earned value, the team can tell precisely where the project stands. For example, the team might report the weekly data shown in Table 2.

Table 2: Weekly Team Data

Week 3	Plan	Actual
Task hours	106	98
Task hours to date	300	274
Earned value	1.9	2.1
Earned value to date	5.8	5.3

Even during the early requirements or design work, the team reports tell management precisely where the team stands against the plan. For example, from the data in Table 2, management could see that the team's performance has improved significantly in the last week. In the previous two weeks, they had 176 task hours ($274 - 98$), or an average of 88 task hours per week. This week they achieved 98 hours. Similarly, even though they are behind on cumulative earned value to date, they accomplished more than planned in the latest week.

The earned value method also helps teams estimate when they will finish the job. From the weekly data, the engineers can see how many hours they have spent for each point of earned value. Then, assuming that they continue to work at the same rate, they can estimate when they will finish the work. For example, from Table 2, the team has accumulated 5.3 earned value points (EV) in 3 weeks, or 1.7667 EV per week. At that rate, the job will take $100/1.7667 = 56.6$ weeks. Since they have completed week 3, this is 53.6 more weeks of work. If the team is able to continue working at the rate for the most recent week, however, the total job would take only $(100 - 5.8)/2.1 = 44.9$ more weeks.

The earned value data not only help team members to see where they are, but these data also help management to understand what needs to be done to complete the job on time. With earned value, TSP teams and their managers can anticipate schedule problems very early in the job. Then, they generally have time to take any needed recovery actions.

While earned value is helpful in tracking team progress and providing the engineers with a sense of accomplishment, it does not address task priorities or dependencies. To properly manage task relationships, the engineers must maintain their personal plans and ensure that they identify and resolve all task dependencies with their teammates. On larger teams, helping the team members do this is an important part of the planning manager's role responsibility.

6.4 Rebalancing Team Workload

Unbalanced workload can cause a team to be inefficient. This occurs when some engineers have much more work than others have. This problem has several causes. First, the most experienced engineers are generally involved in much more of the work than the team members with less experience. While the most experienced engineers could probably do each task faster and better than the others, this would overload them and leave the others with little to do. Another cause of unbalanced workload is the normal fluctuation in engineering performance. Some engineers will finish their tasks ahead of the plan, and others will fall behind.

While unbalanced workload is natural, it is not efficient. Unless every member of the team is always fully occupied, the team cannot be fully effective. Every week, when the team examines project status, the engineers can see if their workload is unbalanced. If it is, the team should rebalance the plan. With the TSP, teams do this as often as needed—every week if necessary. Once the engineers have completed a TSP launch and have detailed personal plans, they can generally rebalance team workload in only an hour or two.

The TSP team launch produces an overall project plan extending from the initial team launch until final project completion. Depending on the project, the plan could cover a few weeks, or it could take many years. With the TSP, every team member produces a detailed plan for the next project phase. Since engineers cannot generally make detailed plans for more than about three or four months, the TSP breaks projects into phases of about three to four months duration. Teams relaunch their projects at the beginning of each phase or cycle. Whenever teams find that the plan no longer helps them to do their jobs, they should relaunch their projects. Teams should also be relaunched when there are major changes in the work to be done or in team membership.

7 TSP Quality Management

While most organizations will agree that quality is important, few teams know how to manage the quality of their products. Furthermore, there are no general methods for preventing the injection of defects. People develop software, and people make mistakes. These mistakes are the source of software defects. In the TSP, the principal quality emphasis is on defect management.

To manage quality, teams must establish quality measures, set quality goals, establish plans to meet these goals, measure progress against the plans, and take remedial action when the goals are not met. The TSP shows teams how to do this. The elements of TSP quality management are making a quality plan, identifying quality problems, and finding and preventing quality problems. These topics are covered in the following paragraphs.

7.1 The Quality Plan

During the team launch, TSP teams make a quality plan. Based on the estimated size of the product and historical data on defect injection rates, they estimate how many defects they will inject in each phase. Where teams do not have historical defect injection data, they can use the TSP quality planning guidelines shown in Table 3. These will help them establish quality goals. Once the engineers have estimated the defects to be injected, they estimate defect removal, again using historical data or the TSP quality guidelines. These removal estimates are based on the yield for each defect removal phase. Here, yield refers to the percentage of the defects in the product at phase entry that are removed in that phase. Once the injection and removal estimates have been made, the team can generate the quality plan. Finally, the team examines the quality plan to see if the quality parameters are reasonable and if they meet the team's quality goals. If not, the engineers adjust the estimates and generate a new quality plan.

Table 3: TSP Quality Guidelines¹

Measure	Goal	Comments
Percent Defect Free (PDF)		
Compile	> 10%	
Unit Test	> 50%	
Integration Test	> 70%	
System Test	> 90%	
Defects/KLOC:		
Total defects injected	75 - 150	If not PSP trained, use 100 to 200.
Compile	< 10	All defects
Unit Test	< 5	All major defects (in source LOC)
Integration Test	< 0.5	All major defects (in source LOC)
System Test	< 0.2	All major defects (in source LOC)
Defect Ratios		
Detailed design review defects /unit test defects	> 2.0	All major defects (in source LOC)
Code review defects/compile defects	> 2.0	All major defects (in source LOC)
Development Time Ratios		
Requirements inspection/requirements time	> 0.25	Elicitation in requirements time
High-level design inspection/high-level design time	> 0.5	Design work only, not studies
Detailed design/coding time	> 1.00	
Detailed design review/detailed design time	> 0.5	
Code review/code time	> 0.5	
Review and Inspection Rates		
Requirements pages/hour	< 2	Single-spaced text pages
High-level design pages/hour	< 5	Formatted design logic
Detailed design text lines/hour	< 100	Pseudocode ~ equal to 3 LOC
Code LOC/hour	< 200	Logical LOC
Defect Injection and Removal Rates		
Requirements defects injected/hour	0.25	Only major defects
Requirements inspection defects removed/hour	0.5	Only major defects
High-level design defects injected/hour	0.25	Only major defects
High-level design inspection defects removed/hour	0.5	Only major defects
Detailed design defects injected/hour	0.75	Only design defects
Detailed design review defects removed/hour	1.5	Only design defects
Detailed design inspection defects removed/hour	0.5	Only design defects
Code defects injected/hour	2.0	All defects
Code review defects removed/hour	4.0	All defects in source LOC
Compile defects injected/hour	0.3	Any defects
Code inspection defects removed/hour	1.0	All defects in source LOC
Unit test defects injected/hour	0.067	Any defects
Phase Yields		
Team requirements inspections	~ 70%	Not counting editorial comments
Design reviews and inspections	~ 70%	Using state analysis, trace tables
Code reviews and inspections	~ 70%	Using personal checklists
Compiling	~ 50%	90+ % of syntax defects
Unit test - at 5 or less defects/KLOC	~ 90%	For high defects/KLOC - 50-75%
Integration and system test - at < 1.0 defects/KLOC	~ 80%	For high defects/KLOC - 30-65%
Before compile	>75%	Assuming sound design methods
Before unit test	> 85%	Assuming logic checks in reviews
Before integration test	> 97.5%	For small products, 1 defect max.
Before system test	> 99%	For small products, 1 defect max.

¹ Note: Only use these initial criteria until you have historical TSP data and can develop your own.

Once the team members have generated the quality plan, the quality manager helps them to track performance against it. The quality plan contains the information shown in Table 4. The quality manager tracks the data on each phase for each part of the system to see if the measures are within the values set by the quality plan. If not, the quality manager raises the issue in the weekly meeting and suggests what the team should do about it.

Table 4: TSP Quality Plan—Form SUMQ

Name	_____	Date	_____
Project	_____	Launch/Phase	_____
Part/Assembly	_____	Assembly Level	_____
Percent Defect Free	Plan	Actual	
In compile	_____	_____	
In unit test	_____	_____	
In integration test	_____	_____	
In system test	_____	_____	
In acceptance test	_____	_____	
In one year of use	_____	_____	
In product life	_____	_____	
Defect/page			
Requirements inspection	_____	_____	
HLD review	_____	_____	
HLD inspection	_____	_____	
Defects/KLOC			
DLD review	_____	_____	
DLD inspection	_____	_____	
Code review	_____	_____	
Compile	_____	_____	
Code inspection	_____	_____	
Unit test	_____	_____	
Build and integration	_____	_____	
System test	_____	_____	
Total development	_____	_____	
Acceptance test	_____	_____	
Product life	_____	_____	
Total	_____	_____	
Defect Ratios			
Code review/Compile	_____	_____	
DLD review/Unit test	_____	_____	
Development time ratios (%)			
Requirements inspection/Req. time	_____	_____	
HLD inspection/HLD time	_____	_____	
DLD/code	_____	_____	
DLD review/design	_____	_____	
Code review/code	_____	_____	
A/FR	_____	_____	
Personal review rates			
DLD lines/hour	_____	_____	
Code LOC/hour	_____	_____	

Inspection rates

Requirement pages/hour

HLD pages/hour

DLD lines/hour

Code LOC/hour

Part/Assembly _____

Date _____

Defect-injection Rates**Plan****Actual**

Requirements

HLD

DLD

Coding

Compile

Unit test

Build and integration

System test

Defect-removal Rates**Plan****Actual**

Requirements

System test planning

Requirements inspection

HLD

Integration test planning

HLD inspection

DLD review

Test development

DLD inspection

Code

Code review

Compile

Code inspection

Unit test

Build and integration

System test

Phase Yields**Plan****Actual**

Requirements inspection

HLD inspection

DLD review

DLD inspection

Code review

Compile

Code inspection

Unit test

Build and integration

System test

Process Yields**Plan****Actual**

% before compile

% before unit test

% before build and integration

% before system test

% before system delivery

7.2 Identifying Quality Problems

In the TSP, there are several ways to identify quality problems. For example, by comparing the data for any module or component with the quality plan, one can quickly see where defect densities, review rates, yields, or other measures deviate significantly for the team's goals.

For even relatively small projects, it can take a substantial amount of time to examine the process data. To help alleviate this problem, the TSP introduces a series of quality measures. These measures are

- Percent defect free—PDF
- Defect-removal profile
- Quality profile
- Process quality index—PQI

A typical PDF plot is shown in Figure 5, which shows the percentage of the system's components that had no defects found in a particular defect removal phase. By tracking PDF curves, one can see if any particular phase is troublesome. This can be seen by comparing the PDF curves for several similar projects. Where there are problems, the quality manager can look at data on lower level components to identify the source of the problem and recommend what the team should do about it.

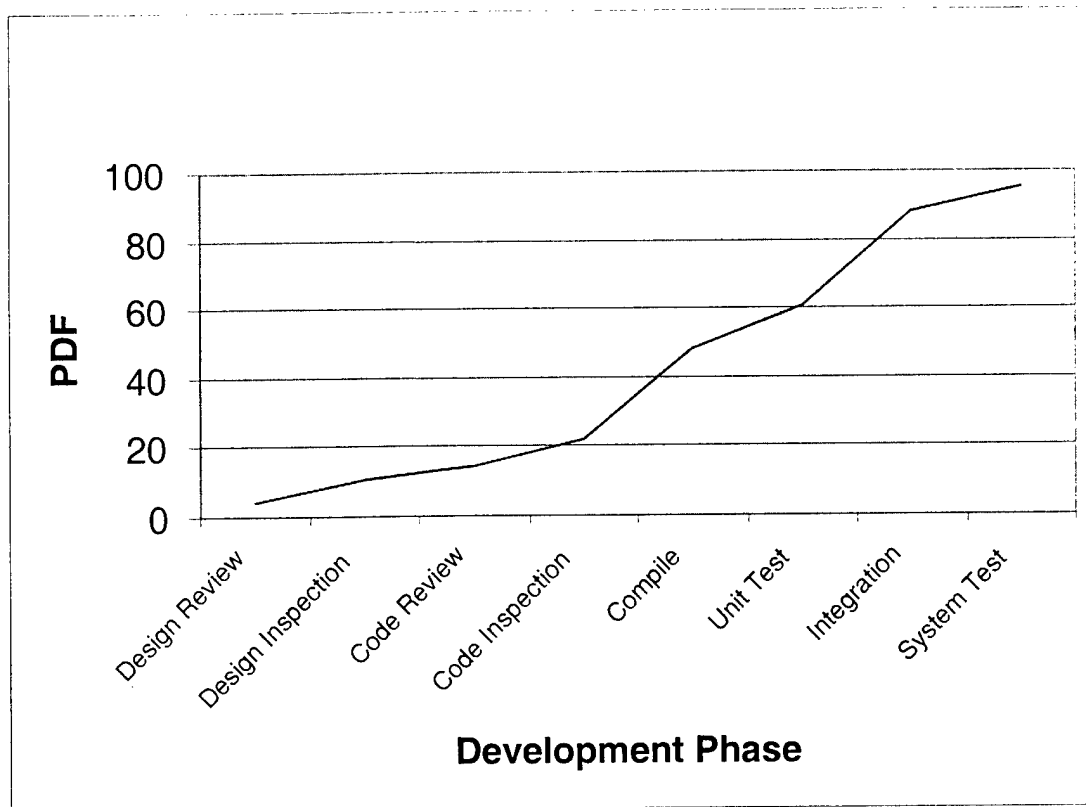


Figure 5: *Percent Defect Free (PDF)*

Figure 6 shows a typical defect-removal profile. While the PDF plot can only be produced for an overall system or large component, the defect-removal profile can be drawn for the system, each of its subsystems, any component, or even down to the module level. Thus, if the PDF or system level defect-removal profile indicates problems, the quality manager can examine progressively lower level defect-removal profiles to find the source of the trouble.

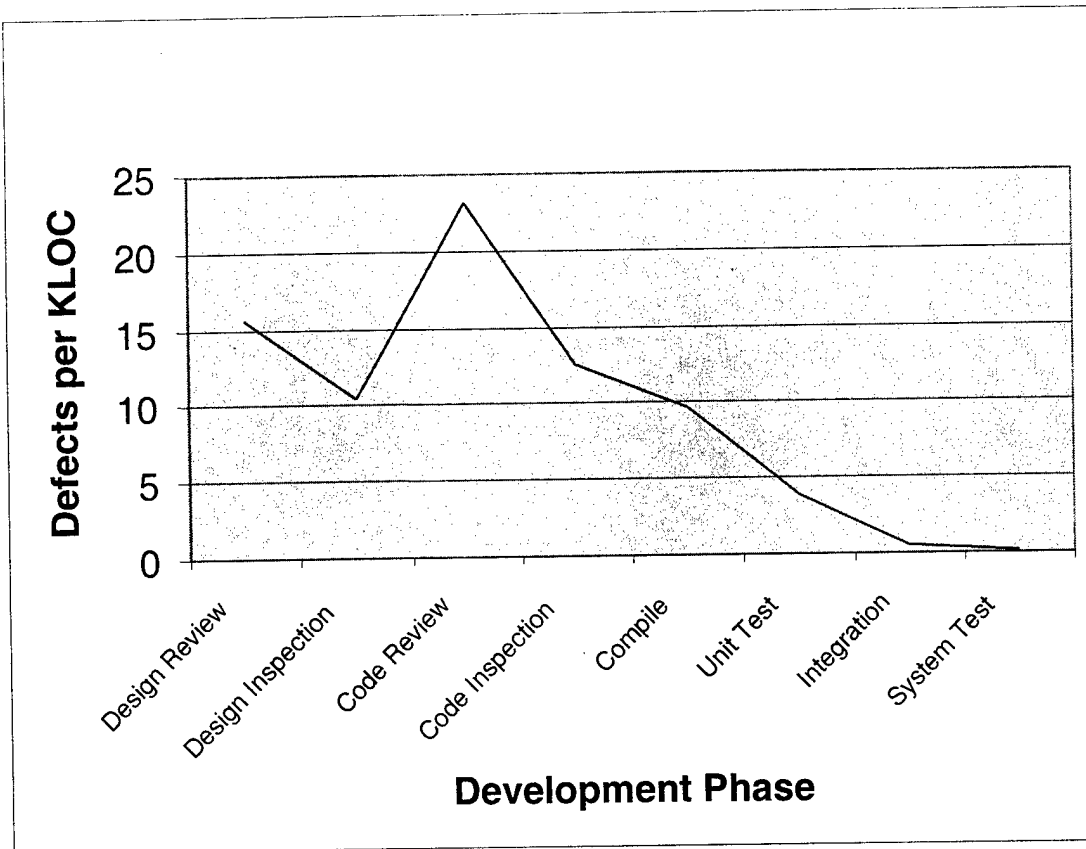


Figure 6: Defect Removal Profile

The quality profile is used with individual program modules [Humphrey 98]. An example quality profile is shown in Figure 7. The quality profile measures the process data for a module against the organization's quality standards. If the organization does not have sufficient historical data to produce its own standards, the TSP quality guideline suggests values. The five quality profile dimensions indicate module quality based on data for design, design reviews, code reviews, compile defects, and unit test defects. After a little practice, PSP-trained engineers can rapidly examine a large number of quality profiles and identify the product elements with likely quality problems.

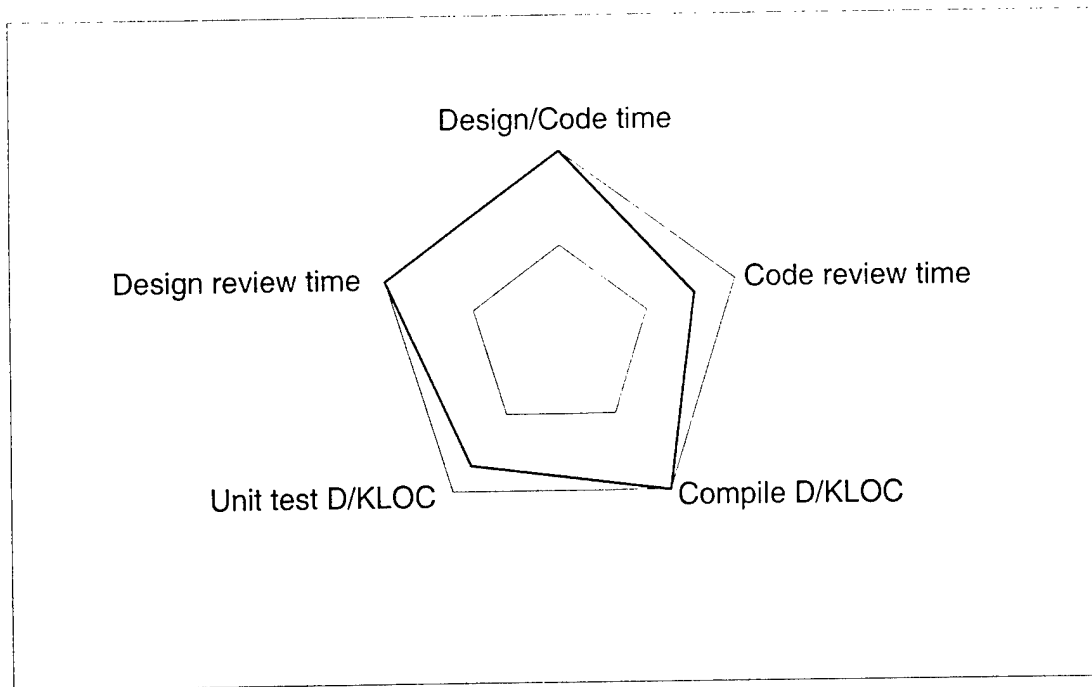


Figure 7: The Quality Profile

The process quality index (PQI) is produced by taking the product of the five dimensional values of the quality profile to produce a single quality figure of merit [Humphrey 1998]. With PQI values above about 0.4, program modules are generally defect free. With the PQI, organizations can sort a large number of modules by their likely quality levels. This is particularly useful with very large systems having hundreds or thousands of modules. Teams can quickly zero in on the modules that are most likely to be troublesome in test or customer use.

7.3 Finding and Preventing Quality Problems

The TSP quality measures can indicate likely quality problems even before the first compile, and they provide a reliable measure of module or component quality before the start of integration or system test. Once a TSP team has identified the modules or components that most likely have quality problems, the remedial actions suggested by the TSP are as follows:

- Monitor the module during test to see if problems are found and then determine the remedial action.
- Reinspect the module before integration or system test.
- Have an engineer rework the module to fix suspected problems.
- Redevelop the module.

The PSP and TSP processes are designed to prevent problems before they occur. In PSP training, engineers typically learn how to reduce their defect injection rates by 40% to 50%. In the TSP, the design manager can further reduce defect injection rates by ensuring that the team produces a complete and high-quality design. The quality plan and process tracking make the engineers more sensitive to quality issues so that they are more careful, reducing defects even further. Finally, the TSP introduces a defect review where every post-development defect is analyzed to identify potential process changes that will find or prevent similar defects in the future.

8 TSP Introduction

The TSP is being introduced into both industrial and academic environments. An introductory TSPi process and textbook are available for teaching university team courses [Humphrey 00]. Before taking the TSPi course, students must have taken a PSP course with either the Discipline for Software Engineering text or the Introduction to the Personal Software Process text [Humphrey 95, Humphrey 97]. The TSPi course has been taught in a number of universities and the early experience indicates that the students and faculty find it helpful. However, no studies have yet been published on the use of the TSPi process and methods.

Introducing the TSP into engineering organizations is the principal focus of the TSP effort at the Software Engineering Institute (SEI). The TSP was designed for engineering teams, and its introduction has been initially targeted at teams developing software-intensive products. To support use by industrial teams that include other than software specialties, the SEI has developed an introductory PSP course for professionals who are not software proficient. It has also introduced a series of training and qualification programs so that organizations can obtain their own PSP instructors. In addition, the SEI provides TSP coach training so that organizations can launch and coach their own TSP teams. The SEI has established relationships with a number of transition partners who are qualified to teach the PSP and to coach TSP teams. (For further information on these topics, refer to www.sei.cmu.edu/tsp.)

9 TSP Experience

While the TSP is relatively new, results have been reported by a number of organizations. Some examples are summarized in the following paragraphs.

- Teradyne found that, prior to the TSP, defect levels in integration test, system test, field testing, and customer use averaged about 20 defects per KLOC (1,000 lines of code, or LOC). The first TSP project reduced these levels to 1 defect per KLOC. Since it cost an average of 12 engineering hours to find and fix each defect, Teradyne saved 228 engineering hours for every 1,000 LOC of program developed. Since the typical cost to code and unit test 1,000 LOC is about 50 engineering hours, the savings in defect repair costs were about 4.5 times the cost of producing the programs in the first place.
- Hill Air Force Base, near Salt Lake City, Utah, is the first U.S. government organization to be rated at CMM Level 5 [Paulk 95]. The first TSP project at Hill found that team productivity improved 123% and test time was reduced from an organizational average of 22% to 2.7% of the project schedule [Webb 00].
- Boeing, on a large avionics project, had the results shown in Figures 8 and 9. The 94% reduction in system test time resulted in a substantial improvement in the project schedule and allowed Boeing to deliver a high-quality product ahead of schedule.

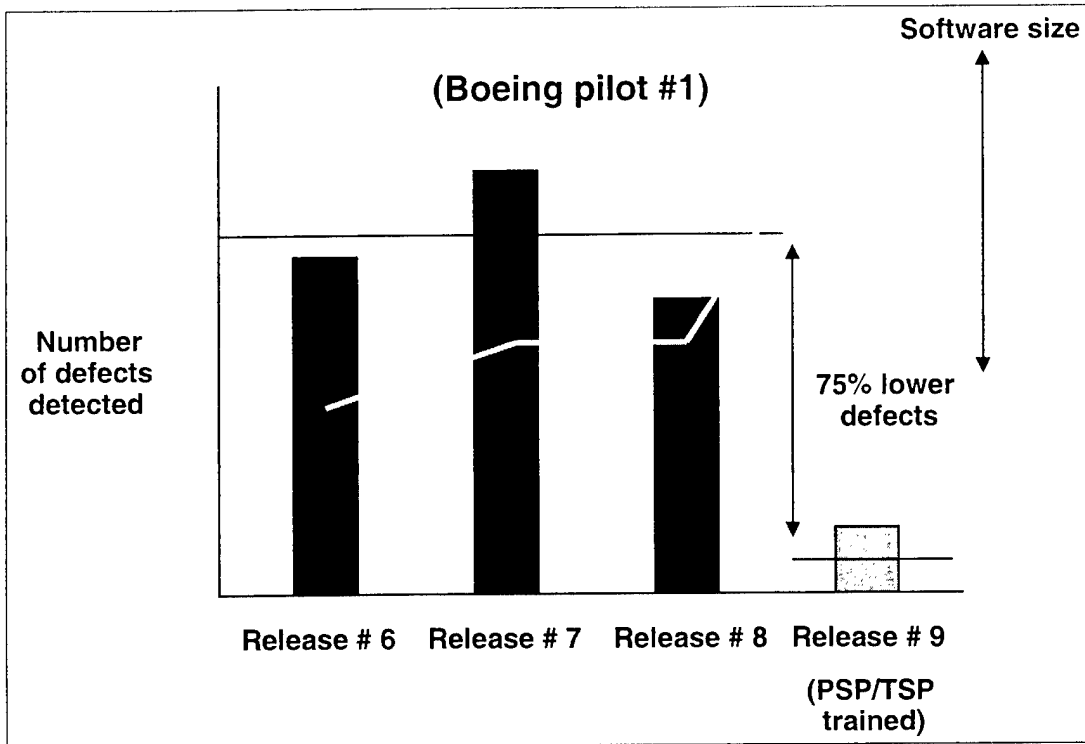


Figure 8: TSP Test Defects

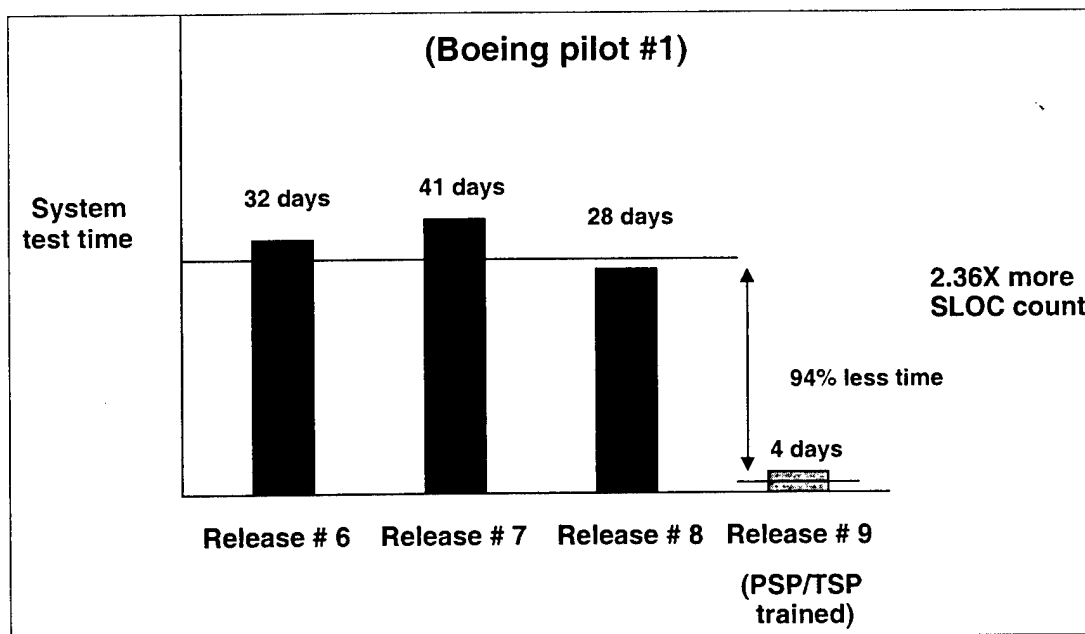


Figure 9: TSP Test Time

10 Status and Future Trends

The initial reason for developing the TSP was to provide an environment where PSP-trained engineers would find it natural to use disciplined methods. PSP training by itself had not been found sufficient to get engineers to consistently use the methods [Ferguson 97]. There are several reasons why this is the case. First, without training, managers generally do not understand the PSP methods or appreciate their benefits. They then often object to their engineers spending time on planning, doing personal reviews, or gathering and analyzing data. Second, disciplined work is hard to do even with support and coaching. Without such help, long periods of sustained disciplined work are almost impossible. The initial motivation for the TSP design was to address these problems.

10.1 Training and Support

TSP training is offered by the Software Engineering Institute (SEI) for team leaders and launch coaches. The principal training needs of engineers are covered by the PSP training courses.

Tool support is also very important for the TSP. The SEI has developed a prototype tool that it makes available to teams it has launched as well as to its transition partners. It is also developing a tool specification to guide commercial vendors in developing tools to support the TSP process. Without proper tool support, the volume of data produced by even relatively small projects is almost unmanageable. (For further information on the available SEI materials and support, see <http://www.sei.cmu.edu/tsp>.)

10.2 Future Trends

Since the initial TSP objectives have largely been met, the current and immediate next TSP development efforts are to transition the basic TSP process into general industrial use as well as to increase the number of academic institutions teaching these methods. The principal focus of the industrial work is on improving the training and introduction methods so that engineers more faithfully follow the process and to encourage the development of commercial TSP support tools and environments. Future activities will include extending the TSP process to various types of teams and to larger teams. In the longer term, extensions are needed for very large teams. The academic-related efforts primarily concern faculty workshops and the publication of results.

Because of the wide variety of teamwork situations, a series of TSP processes will be needed. The basic TSP was designed for teams of 2 to 20 members but it is most effective for teams of 3 to about 12 people. The multiple TSP process, or TSPm, is designed for multiple teams of up to about 100 to 150 engineers. In addition, several TSP extensions will be needed for distributed teams that have members at different physical locations and for large projects with several teams at multiple locations. Similarly, a functional team process will be needed for test or maintenance teams. Finally, a TSP extension will be required for really large teams of several hundred to several thousand engineers who work on large program-wide projects. This process must also span organizational and technology boundaries.

As teams grow larger, the relationships among the PSP, TSP, and CMM will become more important. With truly large program-wide teams, the distinction between the team and the organization process will disappear. Here, the organizational CMM activities must be incorporated into the team process in much the same way that the operational process activities are incorporated at present. With smaller TSP and TSPm teams, the team process and the organizational process must be related. Work is needed to more closely couple these process improvement strategies and to show both the CMM and TSP communities how these two frameworks can complement and support each other.

Ultimately, the relationship of the organizational processes with the overall business process must be defined. Coupling product-related teams with business processes will require some fundamental changes in organizational thinking. However, once these processes are properly related, the ability of TSP teams to capitalize on the skills of their members and to precisely plan and report on their work will significantly improve the overall performance of engineering organizations.

References

- [Cummings 87] Cummings, Thomas G. "Self-Regulating Work Groups: A Socio-Technical Synthesis," *Academy of Management*, Vol. 3, No. 3, July 1978, p. 627.
- [DeMarco 87] DeMarco, T. and Lister, T. *Peopleware, Productive Projects and Teams*. New York: Dorset House Publishing Co., 1987.
- [Deming 82] Deming, W. *Out of the Crisis*. , Cambridge, MA: MIT Center for Advanced Engineering Study, 1982.
- [Dyer 84] Dyer, J. "Team Research and Team Training: A State-of-the-Art Review." *Human Factors Review*, The Human Factors Society, Inc. (1984): 286, 309.
- [Fagan 76] Fagan, M. "Design and Code Inspections to Reduce Errors in Program Development." *IBM Systems Journal* 8, 3 (1976).
- [Fagan 86] Fagan, M. "Advances in Software Inspections." *IEEE Transactions on Software Engineering* SE-12, 7 (July 1986).
- [Ferguson 97] Ferguson, P.; Humphrey, W. S.; Khajenoori, S.; Macke, S.; and Matvya, A. "Introducing the Personal Software Process: Three Industry Case Studies." *IEEE Computer* 30 (May 1997): 24-31.
- [Humphrey 89] Humphrey, W. S. *Managing the Software Process*. Reading, MA: Addison-Wesley, 1989.
- [Humphrey 95] Humphrey, W. S. *A Discipline for Software Engineering*. Reading, MA: Addison-Wesley, 1995.
- [Humphrey 97] Humphrey, W. S. *Introduction to the Personal Software Process*. Reading, MA: Addison-Wesley, 1997.
- [Humphrey 98] Humphrey, W. S. "The Software Quality Profile." *Software Quality Professional* 1, 1 (1998): 8-18.

- [Humphrey 00]** Humphrey, W. S. *Introduction to the Team Software Process*. Reading, MA: Addison-Wesley, 2000.
- [Juran 88]** Juran, J. M. and Gryna, F. *Juran's Quality Control Handbook, Fourth Edition*. New York: McGraw-Hill Book Company, 1988.
- [Katzenbach 93]** Katzenbach, J. and Smith, D. *The Wisdom of Teams*. Boston, MA: Harvard Business School Press, 1993.
- [Mohrman 95]** Mohrman S. *Designing Team-Based Organizations, New Forms for Knowledge Work*. San Francisco, CA: Jossey-Bass Publishers, 1995.
- [Paulk 95]** Paulk, M. et al., *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, MA: Addison Wesley, 1995
- [Shaw 81]** Shaw, M. *Group Dynamics: The Psychology of Small Group Behavior*. New York: McGraw-Hill, 1981.
- [Shellenbarger 00]** Shellenbarger, S. "To Win the Loyalty of Your Employees, Try a Softer Touch." *The Wall Street Journal*, January 26, 2000: B1.
- [Stevens 94]** Stevens, M. and Campion, M. "The Knowledge, Skill, and Ability Requirements for Teamwork: Implications for Human Resource Management." *Journal of Management* 20, 2 (1994).
- [Webb 00]** Webb, D. R. "Managing Risk With TSP." *Crosstalk*, June 2000: 20.

REPORT DOCUMENTATION PAGE

Form Approved

OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. agency use only (leave blank)	2. report date November 2000	3. report type and dates covered Final	
4. title and subtitle The Team Software Process SM (TSP SM)		5. funding numbers C — F19628-00-C-0003	
6. author(s) Watts S. Humphrey			
7. performing organization name(s) and address(es) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. performing organization report number CMU/SEI-2000-TR-023	
9. sponsoring/monitoring agency name(s) and address(es) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. sponsoring/monitoring agency report number ESC-TR-2000-023	
11. supplementary notes			
12.A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12.B DISTRIBUTION CODE	
13. abstract (maximum 200 words) The Team Software Process SM (TSP) guides engineering teams in developing software-intensive products. Early experience with the TSP shows that its use improves the quality and productivity of engineering teams while helping them to more precisely meet cost and schedule commitments. The TSP is designed for use with teams of 2 to 20 members, and the larger multi-team TSP process is designed for teams of up to about 150 members. While TSP versions are planned for larger projects, they are not available at the time of this writing. This report describes the TSP and how it was developed. Starting with a brief background discussion of software quality, the report provides an overview of the basic elements of teamwork. It then describes the relationships among the TSP, Personal Software Process SM (PSP), and Capability Maturity Model® (CMM) process improvement initiatives. The report also describes the TSP process structure, launching a TSP team, the TSP teamworking process, and the issues and methods for introducing the TSP. The report concludes with a review of TSP experience, current status, and trends. SM Team Software Process, TSP, Personal Software Process, and PSP are service marks of Carnegie Mellon University. ® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.			
14. subject terms Team Software Process, TSP, software development, software engineering, quality, productivity		15. number of pages 52	
16. Price Code			
7. security classification of report UNCLASSIFIED	18. security classification of this page UNCLASSIFIED	19. security classification of abstract UNCLASSIFIED	20. limitation of abstract UL